SHERLOCK: A Coached Practice Environment for an Electronics Troubleshooting Job

March 1988

Alan Lesgold
Susanne Lajoie
Marilyn Bunzo
Gary Eggan

LEARNING RESEARCH AND DEVELOPMENT CENTER

University of Pittsburgh

(4)

SHERLOCK: A Coached Practice Environment for an Electronics Troubleshooting Job

March 1988

Alan Lesgold
Susanne Lajoie
Marilyn Bunzo
Gary Eggan

Learning Research and Development Center
3939 O'Hara Street
University of Pittsburgh
Pittsburgh, PA 15260

To appear as a chapter in J. Larkin, R. Chabay, & C. Scheftic (Eds.), Computer assisted instruction and intelligent tutoring systems: Establishing communication and collaboration. Hillsdale, NJ: Lawrence Erlbaum Associates.

SHERLOCK: A Coached Practice Environment for an Electronics Troubleshooting Job

Alan Lesgold
Susanne Lajoie
Marilyn Bunzo
Gary Eggan

Learning Research and Development Center
University of Pittsburgh[1]

Sherlock is a computer-based supported practice environment for a complex troubleshooting job in the Air Force. This chapter describes the training problem for which Sherlock was developed, the principles behind its development, and its implementation. The training problem is severe and representative of a common problem in our high-technology society. People who will fill a position for a brief period (four years or less for many in this Air Force job) carry out a set of routine tasks which are well supported by technology. However, periodically, a breakdown occurs: a novel situation requiring sophisticated problem solving for which little support is available. In many cases, such problems must be referred to an expert, but such expertise is difficult to acquire. Semiautomated, routinized jobs do not afford sufficient opportunities for complex problem-solving skills to develop, so their incumbents lack the skill required to handle novel problems. Sherlock is an environment in which this missing skill can be acquired for a specific troubleshooting job.

The job domain. Sherlock was developed to raise the level of F-15 Manual Avionics Test Station Technicians' troubleshooting knowledge. These technicians repair electronic modules that have been removed from F-15 aircraft because of suspected malfunction. In their daily work, they follow detailed written troubleshooting procedures (part of their Technical Orders) and use a test station. The test station is a large (40 ft^3) system of electronic equipment to which the module being diagnosed can be attached. By setting various switches on the front panels of the test station, the airman can quickly perform tests on the module. When a test in the prescribed test routine isolates the malfunction, the Technical Orders suggest an appropriate repair for the module.

A serious problem arises when the test station itself has a malfunction. Now, instead of following fixed procedures from a check sheet and making use of the substantial technological support of the test station, the airman is much more on his[2] own and must engage in complex heuristic problem solving. Because they may be working in the field far from help, these technicians have to be self-sufficient: they have to be able to repair their equipment. However, because the test station breaks down only about once a month (and even then may be diagnosed and repaired by an expert to minimize downtime), a first-term airman (someone in his first four years of duty) gets few opportunities to learn this most difficult skill on the job. The technical training airmen receive before reporting for work is oriented toward the conceptual knowledge and skill needed for the routine part of the work--which is all there is for perhaps 95% of the time. Yet this leaves the Air Force with a dilemma: its training supposes that test station repair will be learned on the job, but the job doesn't provide the relevant practice opportunities. Sherlock

[1] Direct all correspondence to Alan Lesgold, LRDC, University of Pittsburgh, Pittsburgh, PA 15260. Phone 412/624-7046.

[2] To simplify exposition, we use masculine pronouns. Most, but not all, of our trainee subjects were male.

does (Nichols, Pokorny et al., in press). Less experienced technicians who have practiced on Sherlock about 20 to 25 hours compare in their ability to troubleshoot test station failures with colleagues who have four more years of on-the-job experience.

Sherlock demonstrates a relatively speedy and efficient approach to teaching performance of cognitive tasks where:

1. There is a huge problem space, i.e., there are many possible actions that can be taken, many choices to be made in solving a problem.

2. The job is difficult largely because of its complexity rather than because of some specific conceptual barrier.

3. On-the-job apprenticeship dealing with the difficult aspects of the job is not available.

The nature of the job. To understand Sherlock, it is helpful to understand a bit more about the job for which it was developed. Routine troubleshooting generally is organized around a decision tree (the service manual for most automobiles contains several such trees). At each branching point in a decision tree, one gets a piece of information and then makes a choice. Eventually, the choices lead to a repair action. If that action is successful, the task is completed. A decision tree can be generated any time that one can state a rule for detecting each possible fault in a system, where the rule specifies the conditions under which that fault is present and each condition can be verified by a measurement. The decision tree simply represents an organization of those rules that attempts to minimize the number of conditions that need to be verified before the fault is found. It is a labor-saving artifact.

The test station is another labor-saving artifact. Gathering data takes time. When the data are to be gathered from electrical tests, time is taken up preparing test equipment, reconfiguring the system so that the needed test points are exposed, attaching any required electrical sources, and attaching instrument probes. The test station minimizes these time costs by acting as a large switchboard. By pushing a few buttons and turning a few knobs, a technician can configure a test circuit, specify a measurement to be made, and produce the value of the measurement on a display. Every test has the same basic form: an electrical circuit is created in which there is a stimulus (a source of patterned electrical energy; perhaps a power supply or a signal generator), a device being tested, and a measurement device. In essence, the technician is measuring the effect, at some point(s) in the device being tested, of applying a signal to some other point(s).

When a test "fails" -- that is, when a test result is out of the expected range -- this suggests with high probability that there is a malfunction in the part of the device being tested. The Technical Orders tell what to replace for any given test failure. However, sometimes the mandated repair does not change the situation. This generally means that the intended test circuit was not properly created, the difficult situation that Sherlock deals with. Either something is wrong in the test station, or something is wrong in the cabling between the test station and the device being tested. A false and possibly misleading test value results. If a technician understands what test configuration is not being achieved correctly, then he has greatly constrained the troubleshooting problem. He need only consider the circuit path through the test station that is created for the test that failed, plus the switching circuitry that is involved in producing that configuration. This is because the test station is a giant switchboard. Only the circuit created by particular switching, plus the switching circuitry involved in creating that circuit, need be considered when a circuit fails.

So, an airman technician must learn several things to successfully diagnose test station failures. First, he must be able to figure out what test the station was attempting when it failed; he must have a mental model of the test. From this, he can determine which circuitry, distributed throughout the test station, might be the locus of failure. Then, he must devise a strategy for searching this circuitry for the fault. This will involve narrowing the search space by performing various tests. For example, if no signal even gets to the device being tested, then none can get to the measurement module. In that case, the problem must be between the signal source and the connections of the test station to the device being tested, not in the path from the device to the measurement module of the test station.

Developing the strategy will require some understanding of what a test is, how the test station carries out tests, and how the major components of the test station work. Carrying out a testing strategy will require repeated exercise of certain skills of measurement and of circuit diagram interpretation. So, in addition to a mental model and the right strategies, the technician must have the procedures needed to follow schematic diagrams and to make and interpret tests with various instruments such as an oscilloscope and a multimeter.

Note that a strategy can be systematic but uninformed. For example, one could replace components of the test station one at a time until the bad component was found. This is inefficient because the circuit path involved in a test tends to include a little bit of each of a large number of components. However, serial component-swapping approach is widespread among electronics technicians in general, even though is usually hopelessly impractical.

Principles Guiding Sherlock's Development

In building Sherlock, we were constrained by several principles. Additional constraints arose from the project's status as a prototype effort. In this section, we discuss why Sherlock is the way it is. Sherlock is a practice environment; it affords chances to practice doing a task in a supported context. At the most abstract level, it is the cognitive version of earlier approaches to errorless learning (Terrace, 1964). With support from a computerized coach, the technician successfully completes problems of the complexity he wants to learn to handle on his own.

The role of practice. Rather than calling it a tutor, we refer to Sherlock as a supported or coached practice environment. The fundamental purpose of Sherlock is to provide an efficient artificial means of practicing, with support and feedback, a skill that cannot be practiced very efficiently in its real application context. The routine activity of the manual avionics test station technician is to follow an essentially fixed procedure. Specific rules are read from a document and followed; there is little judgment required. On the other hand, when the test station fails, a strategy must be invented on the spot--a different strategy each time, because the failure is different each time. Verbal principles have to be mapped into richly varying situations in order to carry out this work. Practice is the means for making initially verbal knowledge stronger and more flexible (Anderson, 1983).

What makes practice time effective? For practice time to be effective, several things are required. First, the proportion of time that involves the target cognitive activity should be maximized. If time is spent waiting for parts, dismantling components, and carrying out long sequences of performances that are already well learned, then that time is not available for practice of the decision process that is the core of troubleshooting facility. Second, cognitive overload should be controlled. In a complex cognitive performance like troubleshooting a test station failure, a trainee may simply forget what he has done so far or where he is in carrying out a plan. While trainees need to learn to handle this complexity, we assume that this virtuosity will come with practice. So, we provide help when a trainee loses track of what he has done. This minimizes the experience of inadequacy and saves the time it would take to declare a failure and stage another problem. Often, simply being reminded of what he has already done will be sufficient to enable a trainee to go on to work on a problem effectively. Third, advice should be available to overcome knowledge gaps. If the trainee simply does not know what to do or how to proceed, giving him a hint will both teach the missing knowledge and enable overall problem solving practice to continue. Fourth, performance should be coached beyond success toward optimality. Just as an athletic coach can watch a complete performance and then improve it by providing useful insights to an athlete, so a problem solving practice coach should provide critical insights that move the trainee toward more efficient performance.

Refining the environment rather than refining the teacher. While coaching is an important aspect of Sherlock, primary emphasis is on the practice environment, not on the role of coach. This is a difference in nuance, not an absolute difference, but it means that we are focused primarily on having the environment reveal its structure rather than on having the coach know exactly what to say to a particular student. This approach includes certain biases. For example, the trainee is given much of the responsibility for learning. Very little feedback is provided other than that which the work environment normally provides; help must be requested. The assigned tasks, though at the limits of trainees' capabilities, must all be completed, with support from the environment available on demand. Fidelity to the work environment is emphasized over precision of student modeling. Our view is that Sherlock can know exactly how the solution of a problem is proceeding but has only imperfect data about what a student knows, because inexpert performance is inherently unreliable.

The context-specific abstracted problem space. Diagnosing a test station failure is a big problem, because of the many components the station contains. Having the right mental model constrains the problem, but it is still extremely complex. Sherlock's problem spaces[3] are computationally manageable only because several techniques have been used to make them smaller (cf. Lesgold, Lajoie, Logan, & Eggan, in press). First, they do not contain all the theoretically possible combinations of actions. They contain only actions that actually tend to be taken by trainees or experts. A number of actions that might, in principle, be taken are left out simply because our experts assure us that they never occur. Second, a separate problem space is developed for each problem. While these spaces have a lot in common, the space for one problem is much smaller than the space of all possible test station failures, because only a small part of the test station is involved in a given test, and the context of failure is that some test of an avionics device yielded spurious results. By eliminating actions that are not seen in real life and by specializing the problem space to a particular problem context, the space can be made much smaller, though it also becomes more context-specific.

The third technique used to simplify the problem space is abstraction. For purposes of representing the problem space, we do not consider each of the test station's individual electrical components, of which there are thousands. If we did this, just representing the possibility that each one of the thousands of parts could be broken would make the problem space unmanageably large. So, we abstract the test station structure to the level of replaceable modules, such as printed circuit boards, rather than representing each component on each board separately. As a result, our context-specific abstracted problem spaces have 50 to 60 nodes, rather than the thousands of nodes that a "complete" problem space for test station diagnosis would have.

An object-oriented approach. The individual test measurements that trainees make while diagnosing station failures are measurements at specific points within a circuit, not generic measurements of a whole circuit board. This is handled by having a specialist miniprogram for each node of the abstracted problem space. Computer software designers call these specialist miniprograms objects. A generic template object is first defined, and then each node's "personal" object is specified as a specialization of the generic template. Most of the specialization consists of data, not program: specific hinting information, test point values, etc.

Substantial savings result from this combination of an abstracted problem space with specialized objects to bring each abstraction back to the real world's level of detail. Rather than a network of separate problem space connections between every detail of one module and every detail of each of the others, there is only one path to or from each abstracted object. The approach is similar to the pattern of traffic in a large city. The abstracted problem space is like the network of main streets, and the objects are like the side streets of small neighborhoods. One gets from a neighborhood to a main street, takes the main street to another neighborhood, and then takes side streets to a specific address. Separate roads connecting every pair of addresses would be unthinkably more complex.

[3] As noted in the introduction to this volume, a problem space is a network whose nodes represent states of partial problem solution and whose links represent transitions between those states. Thus, any solution process can be represented as a path through the problem space from the problem as originally posed to a solution.

The goals of training: Curriculum. So far, we have discussed properties of the problem space. A separate concern is the curriculum, the set of instructional goals and the sequence of activities that are meant to accomplish those goals. Sherlock is a holistic practice environment. What trainees do with Sherlock is to solve difficult test station diagnosis problems. Each problem exercises many of the needed skill components in the context of a complete, naturalistic performance. This is very different from many instructional activities, such as courses, where different sessions treat different curricular goals. In Sherlock, every session addresses almost every curricular goal. Nonetheless, Sherlock does have an explicit representation of the instructional goal structure it wants to achieve. The three main areas of its goal structure, or curriculum, are (a) troubleshooting strategy, (b) mental models of test configurations that can be created with the test station, and (c) using test instruments to make measurements.

Modeling the student's emerging general competence and problem specific performance. Intelligent training systems generally maintain some form of student model. Tutors that deal with constrained bodies of knowledge, like proofs of geometry theorems, model the student by trying to define a computer program that will behave just as the student behaves. A comparison between that program and a program that simulates an expert reveals what the student has not yet learned and is the basis for interactions with the student. Sherlock does not use simulation techniques to model student performance. Its diagnoses are made in a simpler way.

Basically, Sherlock keeps two kinds of student models for each trainee. A competence model is maintained; this is a notation of how well each curricular goal has been achieved. A performance model is generated for each problem that a student attempts to solve. It is an annotation of how well the student is expected to do at each point of the abstracted problem space. The object (microprogram) corresponding to each node of an abstracted problem space has, as part of its local microdatabase, information about which curricular goals are relevant to its execution.

Developing a predicted performance model for a student on a specific problem amounts to determining, for each node of the abstracted problem space, how well the student has mastered the curricular goals relevant to that node. The performance model then influences how reticent Sherlock will be in providing explicit help at specific points in the problem space. After the problem is solved, any divergence of student performance from the predictions can be considered in updating the student's competence model. So, if a student does better than expected at some point in a problem, this is prima facie evidence that he has improved on the curricular goals relevant to that point.

An Example

The domain targeted by Sherlock and its mode of operation may become clearer through an example problem. One of our easier problems involves a situation in which, while a device from an airplane is being tested, an incorrect meter reading appears on the test station. The documentation available to the technician asserts that replacing a particular part of the device will result in restoration of the correct meter reading. When this replacement fails to eliminate the problem, the test station itself becomes a possible suspect. An expert will realize that only a small portion of the test station is involved in the test measurement that failed. While the whole test station requires a few hundred pages of schematics to capture all of its complexity, an expert's mental representation of the portions relevant to this problem would fit into one page,[4] shown as Figure 1.

[Figure 1: Mental Model of Expert]

Across the top of the Figure, the path is shown from the device being tested (UUT, for "unit under test") to the digital multimeter (DMM) on the test station. From left to right, the circuit path passes through connectors (J1/P1) to a cable (TP, for "test package") and then through additional connectors (P2/J2) to the test station. Everything else in the diagram is part of the test station. Upon entering the test station, the signal from the UUT passes through three printed circuit boards before reaching the meter (DMM).

4 An extensive task analysis (Lesgold, Lajoie et al., 1986) supported our assumptions concerning expert domain knowledge and performance as well as the ways in which more and less astute technicians handle test station troubleshooting tasks.

These are labeled A1A3A2, A1A3A1, and A2A3A1; the codes reflect three-dimensional coordinates in the 40 ft^3 test station. Each of the printed circuit cards is complex enough to require a page or more of schematic diagrams. In Figure 1, only the relevant connector pins (numbers around the edges) and the path the signal takes through the card are shown. So, for example, the signal enters A1A3A2 through pins 57 and 58 and exits through pins 11 and 12. The top part of Figure 1 represents the signal flow aspects of the problem. However, the signal passes through several relays, which are controlled by switches on the front of the test station. The circuitry involved in translating switch settings into a signal flow path is called the data flow portion of the problem, since data entered as switch settings are translated into a particular station configuration. Four additional printed circuit cards, A2A3A7, A2A3A8, A2A3A9, and A2A3A10, are involved in data flow. Several switches, S52, S53, and S33, generate the signal path control data for this situation by being either open or closed.

An expert solution to the problem is illustrated by the dark ellipses. The expert might first check at the P2/J2 connectors to see if the voltage value expected on the DMM is getting into the test station in the first place [Step 1]. If it is, then he might check halfway between P2/J2 and the meter [Step 2]. In this problem, the signal is found at Step 1 and is missing at Step 2. This implicates card A1A3A2. The expert would verify the card's input at pins 57 and 58 [Step 3]. Since the signal is getting into A1A3A2 but not getting into the next card, A1A3A1, a likely problem is that the relay that switches the signal through A1A3A2 is bad. Step 4 of the expert is to test that relay to see if the actuating voltage is passing through it. Since it is not, the expert traces through the data flow path and notes that the actuating voltage should be coming from A2A3A10. He then checks to see if the correct control voltages are reaching that card [Step 5]. They are not. So, he tests to see if the switch that controls those voltages is working [Step 6]. It shows a short in a part of the switch that should be open [Step 6]. So, the expert swaps the switch [Step 7], and this solves the problem.

In reality, this activity involves perhaps a square yard of complex schematic diagrams, so it is nontrivial. In fact, our trainees generally were far less facile than the experts in solving this problem. For example, while our expert solved it in seven steps, one trainee took 14 steps and another took 20 (the others were similar). Figure 2 shows the activity of the trainee who took 14 steps. Superficially, it is apparent that he tended to trace through circuit paths from one end to the other rather than trying to isolate the problem to one half or the other of the remaining suspect region (we call the expert approach "space splitting"). Also, he tried to make a few measurements that were not on the circuit paths involved in the failure configuration. Some redundant tests were also done (e.g., Step 6 vs. Step 8). However, he did solve the problem.

[Figure 2: Mental Model of Trainee]

While performing tests on card A1A3A2, our example trainee asked for help several times. First he was just given an overview of what he had done so far. Asking for additional help, he was told by **Sherlock**:

> You need to determine if the 0.0 ohms from the UUT is getting to this card. If there is an input you need to test the output. If there is no output you need to determine why the 0.0 ohms signal is not getting through this card.

Asking for even more help, he was given more specific information about which pins on the card were relevant. After doing the required measurements, if he had asked for more help, he might have been told:

> The input to the A1A3A2 is good. The output from the A1A3A2 is bad. If there's no output from this card, then you should conclude that this card, or the data to this card which sets the relays, is the cause of the problem. Since A1A3A2 Pin 36 reads +28VDC, when it should read 0VDC if TPA03 is to be set, Sherlock suspects that you should investigate the A2A3A10 card. [The failure was in a switch, but the A2A3A10 card is the next sensible place to look at this point in the problem.]

The trainee in this example tried, on about a half dozen occasions, to take an unsafe step, usually removing a card without first turning off power to it. Whenever this happened, he received an appropriate warning. He also tried to make some measurements with the meter set incorrectly and got feedback about that problem.

Note that an important property of Sherlock is that it affords opportunities for practice of the complex skill and that it speaks primarily when spoken to. In certain respects, it is a rich environment, and the student, not the computer, is the teacher and is in control. Sherlock exerts control where the environment would, on issues of safety or impossibility. It also tries to cut off long ventures down dead ends, but for the most part the trainee decides what steps to take, when to seek advice, etc. We think this promotes active, conscious learning, and trainees have been willing to take the responsibilities that Sherlock implicitly assigns to them.

Implementation of the System

Overview of the Design Approach

Sherlock's knowledge base has three components: the work environment, the abstracted problem space for each problem, and the curriculum.

The work environment. Sherlock presents a partial simulation of the work environment in which the skills it teaches are ordinarily exercised. The controls of the test station are displayed and are manipulable, and control settings are monitored so that unsafe or inappropriate actions can be pointed out and blocked. Since the basic data-gathering action of troubleshooting the test station is to measure electrical properties at various points in the circuitry, Sherlock allows trainees to make measurements by pointing to the spots in a schematic diagram at which meter probes should be placed.[5] In addition to the measurement devices built into the test station, a hand held multimeter is also simulated as an available device that can be applied to the schematic diagrams, and so is a wire, since many tests can be performed by shorting across various points and observing the effects on overall test station function.

The schematic diagrams are taken from the Technical Orders, printed manuals used by technicians in their everyday work. There are, of course, other components to the Technical Orders besides the schematic diagrams, including indexing of the schematics by test point and componential structure. There are also extensive troubleshooting procedures, for diagnosis of units from the aircraft using the test station and for diagnosis of the test station itself (including the "test package" that connects units to the test station).

The abstracted problem space. If we simply had the trainee carry out tests by pointing to schematics, inferring his strategy and tactics would be difficult. Also, there are a number of special acts, such as tightening connections, swapping boards, and replacing specific components, that cannot be represented by pointing to schematics. For this reason, a rich hierarchical menu scheme is used as the basic means whereby a trainee traverses the problem space. At any given moment, the trainee can, by appropriate use of an action menu, continue movement through the problem space. The data structures to support this are feasible only because we can specify an abstracted problem space that is small enough to be managed (because of constraints based on the problem itself and the stereotypy of trainee and expert performance). In practice, we have found the abstracted problem space idea to be workable, and we expect that extensions to the approach will not require a major change.

Connections between the abstracted problem space and the other two components. There are loose couplings between the abstracted problem space and the other two components of Sherlock's

[5] As will be discussed below, not all test values are available. That is, our work environment simulation is not as complete as it could be (compare to SOPHIE [Brown, Burton, & De Kleer, 1982], for example). The strategy for deciding which data values should be available is a subject of continued research on our part.

data structure, the curriculum and the work environment. The connection between the curriculum and the abstracted problem space makes possible the primary means of individualized coaching in Sherlock, a rich hinting mechanism. In the current version of the tutor, each student encounters the same 34 problems, which take an average of 35 minutes each for them to solve. The primary mode of individualization over the average 20 hours of coached practice is the hints. The average student sees about 95 hints during the 20 hours, and these are highly particularized according to where in the course of problem solution he was when a hint was requested and how well the object corresponding to that point in the abstracted problem space expected him to perform the next required step.

This individualization is possible because each node of the abstracted problem space is represented by a computational object that knows which curriculum goals must be accomplished before a trainee can handle the part of the problem space it represents. As noted above, this allows the object to construct an expectation of the level of trainee capability for its part of the problem space, using the curriculum based student competence model for that trainee. Each problem space object also knows about a variety of hints, both prepared hints and ones that can be constructed for the specific occasion. The choice among hints is determined partly by the specifics of the student's recent activity and partly by the object's performance expectation for the student.

The connections to the work environment have to do with individual tests the airman may want to carry out, by pointing to various points in schematic diagrams. In these interactions with the work environment, the airman is really taking active problem solving steps, so it is necessary for the work environment to, in some sense, be coupled to the abstracted problem space action menu scheme. This would be easy if the step taken by the airman was always exclusively a sensible next step in his troubleshooting plan. However, it is quite possible for an airman to make additional measurements on a particular board just because he has current access to it.[6] We couple the work environment to the abstracted problem space by having small production systems[7] tied to each work environment unit (each schematic). When the airman finishes testing on a schematic, its production system is run to notify relevant abstracted problem space nodes that information relevant to them has been collected.

The Problem Space Representation

As noted above, the trainee's problem solving activity on Sherlock is monitored and guided by a representation of the abstracted problem space for the problem currently being performed. This abstracted problem space reflects a range of both situation-specific and more general knowledge. This includes the architecture for the test station, the function of the tests that can be carried out with the test station, and various strategies, in either weak (generic) or strong (domain-specific) forms. Each of these knowledge components constrains the problem space in important ways.

Influences of the structural model of the test station. The work environment consists of the test station, the unit from the aircraft that is currently being tested, and a test package which connects that unit to the test station (the station is somewhat generic, and the test package contains the specializing, sometimes active, circuitry that enables it to test the specific aircraft unit). Within the test station, there are twelve large drawers -- power supplies, signal generators, switches, and measurement devices --

6 As will be discussed below, not all test values are available. That is, our work environment simulation is not as complete as it could be (compare to SOPHIE [Brown, Burton, & De Kleer, 1982], for example). The strategy for deciding which data values should be available is a subject of continued research on our part.

7 A production is a rule that calls for some operational step to be taken if a particular set of conditions is met. A production system is a program that consists of a set of rules. The program is executed by repeatedly checking to see which rules' conditions are satisfied, executing their actions, and then recycling through the condition-checking process.

containing the various components that enable testing of aircraft units. Figure 3 shows how this structure is represented on the computer screen in Sherlock.

To a large extent, the drawers represent a first cut on a functional decomposition of the test station. If one were told only that the test station contained a defect, it might be reasonable to try to localize the defect first to a specific drawer and then to a specific circuit board or other component (e.g., transformer) within the drawer, just as a physician's first effort to narrow down a diagnosis might be to decide it is a problem of the cardiovascular system, or the digestive system, or some other system, each of which is in somewhat sequestered anatomy.

[Figure 3: Sherlock Basic Screen]

Influences of the mental model of a test. In fact, though, the real diagnostic situation is much more constrained. Test station failures become manifest when the test station is used for a purpose and that purpose cannot be achieved. We have designed Sherlock to emphasize a fundamental mental model, the mental model of an electronic test. In its generic form, a test consists of four components, a stimulus, a device being tested, a load, and a measurement device. The stimulus is a source of patterned energy that passes through the device being tested to the measurement device, with the load being used to dissipate excess energy or to transform the output signal into a pattern within measurable range.

[Figure 4: Basic Structure of a Test on the Test Station]

Figure 4 helps clarify the core mental model. For any given test that the test station is carrying out on a piece of equipment from an airplane, a signal (perhaps one or more voltage levels, perhaps something as complex as a radar signature) is generated by the test station and routed through a switch (called the switching complex or RAG drawer; more or less like an old fashioned phone exchange) to particular circuit paths in the test package and thence to the unit being tested. Various outputs of the unit being tested are routed through the test package to the switch which then routes them to a measurement device in the test station.

The idea is that the test station configures itself to carry out tests. When it fails, it fails in the course of carrying out a test. The failure must be somewhere in the particular configuration of the four components of the test (stimulus, device being tested, load, and measurement device) or in the paths between them. Consequently, efficient troubleshooting should be organized, in part, as an effort to identify the parts of the test station that are filling the four roles in the test that failed, to identify the pathways in the test station that connect those components, and then to split the space of those identified components into subspaces that can be efficiently determined to either include or not include the system fault. Further complexity arises because the real model of the test station must include the switching device that implements connections among particular components to realize a particular test. That is, the paths between the components of a test are created by other test station components, the switching logic.

Influences of troubleshooting strategies. Finally, the abstracted problem space has to take account of the troubleshooting strategies that novices and experts will employ in trying to find a fault. Basically, there are two types of strategies, space splitting and generic cures. By a generic cure we mean a step one can take when no systematic plan is in hand or apparently productive. For example, sometimes a connection will loosen. So, tightening a number of connections will sometimes fix a test station. Tightening a set of connections would be considered a generic cure if it was done without being motivated by the process of searching the problem space systematically for a solution but rather as a default. This has some chance of fixing the problem. However, generic moves often are a form of thrashing around by novices who do not know what to do.

More interesting are the space splitting variations that we see in domains like the one we have studied. In our samples of airmen who have been through electronics training and who were selected to have high promise of being able to do their jobs, almost all try to be systematic either by space splitting or by moving sequentially through a circuit. What differs with expertise is the space that is split when space splitting strategies are employed. Some split the space of swappable drawers, printed-circuit cards, and other components, while others split the space of functional subdivisions of either the test station in general or the specific test configuration that failed. Even when splitting the space of a functional mental model, details of the situated environment play differing roles in space splitting strategies. We have seen, for example, cases where the space that is split is the model of the test that failed but where the subdivisions are swappable cards. This would work well if the swaps could be done quickly, but in real life (as opposed to our tutor), unmotivated swaps are very inefficient compared to making a few measurements and figuring out which specific component should be replaced.

We do not completely capture these strategy considerations in our tutoring, but our abstracted problem spaces seem sufficient to allow them to occur and to be noticeable. We see little evidence that there were airmen who could not try what they wanted to try in problem solving because of some missing pieces of the problem space. This is partly because the mechanisms that allow specific tests to be done on particular test points of a particular printed circuit board are part of the work environment simulation and not part of the problem space representation per se. At the more microscopic level of placing meter probes anywhere on any of our 50 pages of schematic diagrams, there were occasional cases in which the trainee could not do exactly what he wanted to.[8]

[Figure 5: Abstracted Problem Space for a Sherlock Problem]

<u>What does the abstracted problem space for a problem look like</u>? Figure 5 shows a graphical representation of the abstracted problem space for one of our problems. Each label represents a node in the abstracted problem space, and the lines represent hierarchical relationships; nodes on the left subsume nodes on the right to which they are connected. That is, the nodes branching off to the right of a given node represent the wherewithal for "completing" the purpose of their "parent." This does not mean that every "offspring" of a node must be exercised before that "parent" is completed, though. Sometimes the actions and outcomes associated with a single "offspring" make it clear that a whole region of the problem space can be discounted.

Each node of the abstracted problem space is a computational object with specific data and procedures for handling the variety of circumstances that involve the part of the problem space it represents:

(a) how to keep records of a trainee's activity when he reaches that part of the problem space;

(b) how to provide hints to the trainee when he reaches that part of the problem space;

(c) how to take account of the possibility that action elsewhere in the problem space may have ruled out this object's part of the space; and

(d) how to assure that actions taken in this object's part of the space meet certain requirements for safety, efficiency, and possibility of being performable in the real world.

Table 1 shows an outline of what one of our abstracted problem space objects looks like.

---

[8] We did find that on some occasions an airman would have no trouble moving in our abstracted problem space but would not always be allowed by the tutor to perform the exact test that he wanted. This was partly an artifact. We required airmen to specify which board they wanted to test before going to the simulation to place the probes of their simulated meters. Sometimes, after doing one test, an airman would try to do another on a part of the schematic currently being displayed that was not part of the same board (board outlines were shown in the schematics, of course). This was not accepted by the tutoring system. A more serious problem was that in a few cases airmen tried to do tests that had not occurred to our experts as being likely to occur. This problem is fundamental with respect to the adequacy of abstracted problem space approach and is addressed later below. In future versions, we expect to do a better job of preserving a sound abstracted problem space approach at the more detailed level of the work environment simulation as well as at the menu-driven problem-action-step level.

[Table 1 about here]

Why have a separate problem space representation? As one would suspect, and our data appear to confirm this, there is much more idiosyncracy at the level of individual electronic tests or measurements than at the level of the replaceable components such as printed circuit cards. Thus, the design approach of building individual problems around an abstracted problem space representation but situating the activity of the trainee in the work environment has important practical advantages. For the level of detail at which we have good understanding of expert and novice behaviors, we can be very top-down and plan for how we will want to coach performance, attending to both global and more local details of expert problem solving within the domain. At this abstracted level, coaching can be an intelligent process. For the microscopic level at which our incomplete understanding and combinatorial complexity make this less possible, we can still provide a rich device simulation and a more symptom-driven and preprogrammed level of coaching. With this two-prong strategy, we feel that we can provide a very cost-effective approach to tutor design, including intelligent coaching where this is feasible, and can retain maximal situated detail to insure transfer to the real job environment.

The Work Environment Representation

The work environment has several components. First, there is the representation of the external work environment, including the front panels of the test station, the test package, and the unit from the airplane that is being tested on the test station. When the display shown in Figure 3 is on the screen, the front panel of any of the twelve major test station components can be accessed by selecting (with a mouse) the appropriate one of the twelve boxes shown inside the test station representation. For example, Figure 6 shows the front panel display for the oscilloscope drawer. Each of the knobs and switches is manipulable. Mousing[9] alongside any knob or switch causes it to reset to the position the mouse is pointing to. The effects of the change are propagated to all displays immediately. So, for example, the Channel 1 Volts/Division display in the top center of Figure 6 could be changed from 1 V/Div to 2/Div by mousing the 2 just below the current setting of 1. The result of doing this would be to change the scale of the display, causing the waveform shown on the left to shrink to half its current size.

[Figure 6: Oscilloscope Front Panel Display]

A second part of the work environment simulation consists of schematic circuit diagrams for many, but not all, portions of the test station and test package.[10] These diagrams are also available to the trainee in printed form in a booklet that is at the computer workstation for use during practice sessions on Sherlock. The reason for putting them on the screen is to provide a highly real-world-situated environment for practice. Trainees make measurements on the test station by calling up a schematic and pointing to the test points in the schematic at which meter probes should be applied.

[Figure 7: Making a Test on a Schematic]

Figure 7 shows part of the screen during such a test. The hand-held Simpson™ multimeter is indicated in icon form on the upper left of the Figure. By first mousing the word Red and then mousing a point in the schematic diagram, the trainee can indicate where the red probe should be applied. The icon for the red probe is actually shown in Figure 7, near the middle of the schematic diagram, between test points numbered 52 and 20 (it is covering the test point to which it is applied). It is also possible for the trainee to access the front of the hand held meter to switch it to different ranges and measurement types (ohms, DC+ volts, DC-volts, or amps).

---

[9] By mousing, we mean pointing to an object and then pressing a button on the top of the mouse (pointing device).

[10] The test package is the connector between the test package and the aircraft device that is being tested. It consists of complex cabling and a small amount of circuitry.

There are about 60 schematic diagrams available on the screen. An extensive index is available to the trainee both in print form and actively on the screen (mousing an index entry causes the figure it names to be displayed). In addition, cross-reference tables enable the system to know in which schematics a given test point can be found (in some cases, there are multiple views, and the same test point may be in several schematics, so keeping track of what the trainee is doing requires this kind of cross referencing). Thus, the printed documentation used on the job is well represented in the work environment simulation, simultaneously also allowing a view of the innards of the test station and test package.

The final aspect of the work environment, already mentioned, is a collection of measurement devices for testing hypotheses about the location of the fault in the test station. One device, the hand-held meter, is quite an obvious choice. In addition, the meters in the test station itself can sometimes be used to make measurements on other drawers. However, in order to capture the forms of activity in the real job environment, another device is also provided, a wire. This is because it is often more efficient to short out two test points and observe meter readings on the test station than to take a hand-held meter and trace through a circuit. The wire as a "measurement device" is another example of our extensive efforts to capture the cognitive situation of the real work environment as much as possible. Figure 8 shows the screen after the airman has received hints concerning a move that uses the wire--he ends up performing the test and then asks for and receives additional advice on what the results mean: "This means that the measurement path to the DMM [digital multimeter drawer of the test station] is bad."

[Figure 8: Screen Showing Measurement Made After Getting a Hint]

Implementation of the Work Environment Representation

Drawers and test equipment. The external appearance of the work environment primarily involves the front panels of the test station and the test equipment available for troubleshooting (e.g., the hand-held Simpson™ meter). Each of those panels is represented as a computational object. The object for a drawer or piece of test equipment contains a variety of variable "slots" that store the state of the object and also information about the trainee's use of the object (currently, we store the time and nature of every knob movement in the appropriate object; tests (placements of meter probes) are stored separately, as discussed below). Other slots contain the graphic display details for the background of the object, which remains static, for meter displays, which can change value, and for knob settings, which can also change. Information about how to configure a small iconic representation is also stored for drawers that have meter displays.

In addition, there is associated with each display panel a list of active value regions, screen regions that respond to mousing by changing the display. When a trainee mouses an active region in a drawer display, the change associated with that active region is executed. A common change is to move the pointer on a knob to the location moused, which also involves propagating the new knob setting value to other display components that might be affected by it (e.g., turning the knob that sets the scale of the oscilloscope requires not only changing the knob display but also updating the oscilloscope waveform display).

Each drawer or test equipment object contains specifications, called methods, for carrying out certain procedures. These include methods for updating knob settings and display dials, building the original display of the drawer, and building and updating the smaller iconic display. While the overall program is quite large, each individual updating function is quite compact and manageable. One of the more complex methods deals with settings on continuously moveable knobs. Here, there are two active regions for the knob (one region on the left to indicate decreases and one on the right to indicate increases). Thus, the necessary geometric calculations must be made to decide the angle from the center of the knob to the point moused, redraw the knob setting, and interpret the setting angle in terms of the scale of measurement to which the knob is calibrated.

The schematic diagrams. Another critical part of the test station work environment representation is the set of schematic diagrams. In the real work environment, these schematics are part of the Technical

Orders, a multivolume documentation set. In our task analyses, there was some indication that finding material in the Technical Orders was difficult for some first-term airmen, so we wanted to capture the access problem in our work environment simulation. We did this by providing an indexing scheme for screen-based Technical Orders pages that corresponds closely to the indexing scheme in the printed Technical Orders.

There are two important characteristics of the Technical Orders that should be noted. First, we made paper copies of Sherlock's Technical Orders available to airmen while they used the system. Schematic diagrams are very detailed, and the higher resolution of the print medium makes them easier to examine in that form. Second, we developed our own slightly simplified Technical Orders for the tutor. We did this for logistics reasons -- the complete real ones did not become available quickly enough. In retrospect, this may not have been an ideal arrangement, but the success of Sherlock has been demonstrated with post-training criterion tasks that used the real Technical Orders, so not much has been lost due to our simplifications.

The reason for also having the schematics available on the computer is that they were the communications medium by which the airman indicated the points to which he wanted to attach meter probes to make measurements.

Two kinds of indexing were involved in managing Technical Orders content in Sherlock. First, there was a reproduction of the index scheme found in real printed Technical Orders. In any page of this index that appeared on the screen, one could access an index entry by mousing its page number. Second, because of the complexity of the schematics and their partial redundancy,[11] it was necessary for Sherlock to have an index of test points. This index made it possible for airmen to determine which page(s) of schematics they should consult in order to access any given test point.[12]

Carrying out tests. To carry out tests,[13] then, the airman would first indicate in the planning menu that he wanted to test a particular printed circuit board or other component (e.g., a switch). Sometimes, the desired component first had to be "extended," manipulated to make test points accessible. Then, in most cases, he would need to set the controls on a hand-held multimeter, the most common piece of test equipment. The hand-held meter display is another front panel knob-and-dial simulation of the sort shown in Figure 5. After making the appropriate settings, the airman would then call up the appropriate page of schematics, the page on which he could indicate where the meter probes should be placed. The hand-held meter then appears above the schematic diagram in iconic form, as shown in Figure 7.

If the requested test has a result that Sherlock knows about, the appropriate information is displayed in the meter icon, as shown in the Figure. If not, then the airman is informed that the measurement is not available. Sherlock is also able to note cases where the meter is set inappropriately. For example, there may be a spot where one can appropriately measure resistance but where a voltage value might be meaningless, or the meter might be set for DC+ Volts and the probes placed to produce a negative voltage reading. In this case, appropriate comments are offered by Sherlock, so that the airman doesn't think his decision about where to measure was wrong.

11 A given point in the circuitry may appear on more than one page of the schematic diagrams.

12 Test points are denoted by a combination of the printed circuit card number and the specific connector on that card that is desired. So, to access Test Point 2 on Card A1A3A10, an Airman could look up A1A3A10-2.

13 The discussion which follows concerns tests carried out after a fault is evident. The routine part of the job, which involves using the test station to test units removed from aircraft is also simulated in Sherlock, but in a different way. Those tests tend to involve switch settings on the front panels of the test station, while tests carried out to diagnose an apparent test station failure involve manipulation (e.g., placing meter probes) within the circuitry.

In the current version, we relied on expert guidance concerning which test values Sherlock should know about. We provided all test values that our experts believed would be used by experts, along with a number of additional test points that the experts expected trainees to request. Some test points were intentionally omitted, in order to force the trainee to ask for help rather than succeed with an inefficient strategy. For example, it may be possible to find a gap in a lengthy circuit path by exhaustive point-to-point continuity testing starting at one end of the path and working toward the other. In order to avoid having the trainee successfully solve the problem in this way, some test values consistent with the sequential strategy but inconsistent with a space splitting strategy were omitted, making the strategy unworkable.

While this approach was generally successful, some trainees who used the tutor complained about "missing" test points (sometimes because they felt that a purely sequential testing strategy was appropriate), and indeed a small number of reasonably requested test values were not available. We now believe that the key to improving this part of Sherlock is to split the test value matter into two parts. First, all potentially relevant test values should be included. Second, intelligent coaching should be provided in cases where the pattern of tests shows successful but inefficient testing.

The procedure we hope to use in our next version for generating test point values will be based on the notion of a circuit path that is active in a given test station configuration. That is, when the test station is being used to perform a given test on the unit from the aircraft, a path is switched into place involving a stimulus source which is routed to the unit and a measurement device which is also routed to certain outputs of the unit. All of the measurements between consecutive test points in that path should be provided, we now think. Further, if the switching logic is bad or the controls for it are bad, then measurements of points in the paths controlling the point at which the primary test circuit is disrupted must also be included in the collection of known test values. Similarly, paths involving power sources for devices in the primary path must be included if implicated. This would have the effect of making the absence of a test value an indication that it was not relevant to a reasonable model of the test station as configured at the time of failure. Currently, the absence of a test value indicates that it is not relevant to an <u>efficient</u> diagnostic strategy.

To accomplish the second goal, of intelligent coaching for inefficient testing sequences, we will need to include an assessment of testing efficiency in a wrap-up discussion that we would like to add to Sherlock. An intelligent coach might consider the length of the trainee's testing sequence, compare it to the length of an expert sequence, and then comment on the difference if it is large. Similarly, a lucky guess, which now produces only positive feedback, might be labeled as such in a wrap-up commentary.

<u>Connecting the work environment to the abstracted problem space-based coach</u>. We have thus far described two levels of function in Sherlock. The abstracted problem space level is a hierarchy of plans and actions down to the level of actions involving replaceable components: boards, switches, etc. Most of the hinting and virtually all of the student modeling is driven from this level. The work environment represents a more detailed level, the level of individual tests of small aspects of the replaceable components. At this level, coaching must be more generic or must be the result of data sent back to the abstracted problem space level of representation. We now turn to the manner in which the two levels are linked.

At the start of a problem, the student is following the Technical Orders (TO), doing specific tests of the unit from the aircraft by setting controls on the test station and recording meter readings from it. He continues doing this until he finds a test value that is outside the acceptable range, after which he comes under the control of the abstracted-problem-space coach. The tests are conducted using a printed set of Technical Orders which specify which switches must be set on the test station and what the appropriate test values are. Each test in the TO is represented by a specialized object that knows what settings are required on the test station, what value should be displayed on which device to indicate the outcome of the test, what button on the test station must be pushed to produce the reading, whether it is in bounds or not (i.e., whether the test is passed by the aircraft unit or not), how the student has performed on this test if he has already done it before,[14] and a few other details. The object knows how to decide whether doing the test is currently appropriate, how to let the student conduct the test, how to comment if steps are left out or an unsafe condition created, and how to provide a resultant meter reading.

When a fault is first evidenced,[15] the trainee begins interacting with a wider-ranging menu structure that permits him to express plans and take high-level actions, swap replaceable units, reseat boards, etc. Once he proposes to test a board or similar unit, he is permitted to call up the relevant schematic diagrams and choose a measurement device. He then comes under the control of the work environment simulation, doing tests and receiving feedback (but still having access to hints). He continues doing tests as long as he likes, eventually indicating that he is done by mousing a "Done" box on the screen.

At this point, the work environment executes a small production system that is able to update the abstracted problem space by marking any parts of the space that have been ruled out by any combination of earlier abstracted-problem-space moves and work environment tests. Not only are the nodes that are ruled out marked as such, but a message concerning why they are ruled out is sent to them so that they have it available to provide later explanations. Thus, if a trainee later tries to search a ruled-out portion of the problem space, he will receive a message telling him how the proposed action has already been ruled out. Currently, this message is provided after the action is taken, just to advise that it wasn't really necessary. For example, in one of our problems, one trainee did some tests that ruled out the unit from the aircraft as the source of a fault. Later, if he had swapped a card in that unit, this would have been permitted, but he then would have received the message:

> You took a voltage reading at A2A3A7 Pin 58 with respect to ground.
> The result of this measurement ruled out this unit.

This production system linkage is necessarily an expert system. That is, it only contains the rules experts give us, since we acknowledge at the outset that deep reasoning at the level of individual circuit values is an intractable task for a system of the complexity of the test station for which our tutor is designed.

Student Modeling

Hints can be tailored to a student's ability, provided that the coach knows something about that student's ability. This knowledge is called a <u>student model</u>. As discussed above, there are two levels of student modeling in Sherlock. A <u>competence model</u> records the extent to which various instructional goals of Sherlock have been achieved for a given trainee. This model cumulates over the entire experience of the trainee with Sherlock. A second model, the <u>performance model</u>, is computed for each trainee at the time that he starts a particular problem.

Both models are <u>overlay</u> models. That is, they involve a set of goals or actions for each of which an estimate of student capability is recorded. The competence model is based on a curriculum hierarchy which lays out the goals of Sherlock. For each goal or subgoal, a notation is made concerning the level of capability the student is currently thought to have. Four levels are used: <u>unlearned</u>, <u>perhaps</u>, <u>probably</u>, and <u>strong</u>. A capability at the <u>perhaps</u> level has been apparently manifested recently but is thought to be very fragile. If it fails to be manifested in the next few cases where it is appropriate, it will revert to the <u>unlearned</u> state. If it is remanifested, it will move to the <u>perhaps</u> state. If it is regularly and reliably manifested, it will move to the <u>strong</u> state. In addition, Sherlock also keeps the history of evidence used to determine the state for each curriculum subgoal.

---

[14] Normally, a test is repeated after a swap, to be sure that the system has been fixed, so multiple records of student performance on a test may be recorded.

[15] The trainee actually starts in the mode of using the test station to diagnose parts from an aircraft. This part of Sherlock will be described below. We are concerned in the present section with what happens once a fault is evidenced.

When a trainee begins a problem, a performance model for him is computed. For each node of the abstracted problem space, an estimate is made of how well he is expected to do at that node. This estimate is computed by the node object in the following manner. First the node object takes note of which curriculum subgoals are relevant to performance at this node. These objects are listed in a "slot" of the node object. For each subgoal on the list, the student model entry for that subgoal is noted. Based on these entries, expected trainee performance on the node is recorded as good, okay, or bad. After the problem is completed, Sherlock checks to see whether actual student performance at each node matched expected performance. Exceptions to expectations result in changes to the competence model for the student.

Hints

The coaching provided by Sherlock consists primarily of hints to the trainee as he tries to solve each fault isolation problem. Some hints are generated dynamically (especially the recapitulation hint described below), while most are "canned." Except in a few specific cases, hints are provided only in response to a trainee's request. The primary exceptions are when the student has finished taking an action in a problem-space node and now wants to move to another point in the problem space that is definitely a wrong choice. On certain occasions, in the judgement of our expert consultants, certain choices were so clearly inappropriate that it makes sense to tell the trainee this rather than letting him discover it for himself. In these cases, making a bad choice of what to do next prompted a hint explaining why it was a bad choice.

A second kind of case is when the trainee wants to redo a step that has already been explicitly taken, such as swapping a board again.[16] In this case, Sherlock will interrupt with a message that the proposed activity has already been completed. A related case gets different treatment. If the activity associated with a node in the problem space has been rendered redundant because some combination of other actions has logically ruled it out (e.g., several tests of other components might produce results that logically entail the correct function of a different board, which one would then have no reason to swap or test), then the trainee is still allowed to do the activity, but after he is done with it, he is advised that it was not necessary, and told why.

The remaining hints are provided in response to requests for help or pressing the panic button. When the panic button is pressed, the trainee is given a top-down overview of the portions of the problem space that are still candidates for additional activity. Then, he is given the initial planning menu that is available at the time a fault is first encountered, essentially being allowed to start over again, from the top, on what is left of the problem. Our thought was that the one occasion on which we could probably get the trainee to step back from the details of tactics to look at strategy broadly was when he was completely at a loss for next actions.

When a hint is requested by mousing the Help button on the screen, then the system makes a decision on how to respond based on the performance model of the student for the problem space node at which he is currently located and his activity status within the node. There are four categories of activity: Action, Outcome, Conclusion, and Option. Action refers to deciding which test to carry out and how that test is done. Outcome refers to getting the results of a test, perhaps by reading a meter. Conclusion refers to the determining the meaning of the results, and Option refers to deciding what to do next. So, for example, one might enter the problem space node for testing a particular printed circuit board but not know what to measure on that board. Asking for help at that point would produce an Action hint. On the other hand, if one has just made an oscilloscope measurement, then presumably the needed help is in the area of Outcome or Conclusion. If one has already closed the page of schematics for the board and then asks for help, an Option hint is provided.

[16] In Sherlock, all replacement components always work. This is not entirely realistic, but is part of the general strategy of focusing trainee time on activity deemed most likely to improve basic troubleshooting skill.

Several structural details can help assure that an appropriate hint is provided. First, some stages of activity within a problem space node are well marked, especially the Action and the Option stages. Second, as available hints at one level are exhausted, hints for the next level can be provided. So, if one gets some Action hints and keeps asking for more help, eventually Outcome, Conclusion, and Option hints would also be provided. We planned a third approach, in which Sherlock was to put up a menu asking the trainee what kind of help was needed, but this plan was not implemented, due to scheduling urgencies.

Once the type of hint to be delivered is determined, then a decision about how explicit a hint to provide must be made. Hints are classified from 1 to 5 on a scale of explicitness. Hint level 1 is always a recapitulation of what the trainee has done so far in trying to solve the problem.[17] Hint levels 2 to 5 are assigned to pre-written hints that were produced by our subject matter experts. There is not always a hint available for each level and each type.

The first time a trainee asks for help on a node, he gets a Level 1 recapitulation hint. The next time, he gets a hint determined by his expected performance on the node at which he is currently located. If the performance model says that he should do a Good job, he next gets a Level 2 hint. An Okay rating produces a Level 3 hint, and a Bad rating gets a Level 4 hint (see the discussion of the Performance Model above). Further requests result in progressively higher level hints till the appropriate set is exhausted, followed by hints from successive type groups (e.g., if Sherlock is out of conclusion hints, it gives an option hint).

Some General Issues

Situated learning. The ideal of "situated learning" is currently fashionable, especially in informal discussions of instructional systems by their designers. It has been noted (Rogoff & Lave, 1984; Jordan, 1987) that in simpler, more unified and coherent cultures, much of learning is socially supported by groups of people in direct contact with the parts of the environment in which the knowledge will be needed. People learn about details of agriculture by watching planting, cultivating, and other processes and helping with them. Tailors learn to make clothes in an apprenticeship. Often, children follow in their parents' footsteps, and many jobs are done at home, or otherwise with children present, so one lives in the environment one will act in, and one watches and helps with expert actions. Learning does not seem to be a problem when all of these conditions are met.

Life in a large, pluralistic, high-technology society tends to lack all of the characteristics just listed, and there have been proposals for forms of "cognitive apprenticeship" that attempt to provide some of the support that is automatically present in more primitive societies (cf. Collins & Brown, 1988). We have tried to design Sherlock with these ideas in mind. So, for example, the forms of work match those on the real job. Each front panel with knobs and dials looks like the panel in the real work environment. The schematic diagrams, instructions for testing units from aircraft, and other documentation, while slightly simplified, retain many aspects of those used in the real job: the symbols in diagrams are the same, the layout of testing algorithms is the same, the indexing schemes are the same, etc.

What is not yet evident is whether this approach matters. The experiments have not yet been done to compare our approach to alternatives. There are broad alternatives, such as componential rather than holistic training. In addition, there are a number of details of realism or abstraction that might or might not be important. For example, tests might be requested via a menu or calculator keypad in which test point numbers and measurement details (volts vs. ohms, etc.) were requested. We might not require people to put the simulated test station into a safe condition via knob settings before making measurements. We might even go to diagnosis of arbitrary circuits different from those used on the job. Transfer studies will have to be conducted to fully justify our approach. For now, all we know is that, in general terms, it works

[17] It differs from a Panic hint in that it is a structured account of the actions taken, while the Panic hint is a structured account of the problem space portions remaining to be searched.

very well and that airmen are able to start using Sherlock with minimal introduction and without having to invest much thinking capacity into figuring out how to talk to it.

Individualized coaching. A second area worthy of comment is individualization. In most individualized instruction systems, and even in the early plans for ours, individualization is via rate of progress through a curriculum. There is an implicit ranked listing of learning tasks, and people are passed over various of these as a function of the system's knowledge of their competence. We took a different tack. All of our trainees get the same series of problems. What differs with differing competence levels is the specificity of support (through hinting) that they receive. A fast learner will characteristically be given very general and sketchy help unless he keeps asking for more, while a slow learner, at points where he is out of his league, will almost be walked through the problem solving process.

This approach arose after one of us spent a week evaluating uses of individualized print and computer technology at schools in Israel.[18] He observed schools in which there were three different sets of reading material, but all with the same stories in them. The fast track received versions of the stories that were longer and more detailed, and the corresponding workbook assignments were also more sophisticated and demanding. At the slow end, the stories were mostly gist, with simpler words and simpler workbook tasks. Nonetheless, in class discussions, the whole class could talk about the same story and its implications.

We similarly thought that one way to leverage the training provided by Sherlock was to provide common experiences, so that discussions at the end of the day could be about problems and how they were solved rather than about who was ahead of whom. We do know that Sherlock generated a lot of excitement at the bases where it was tested, and we like to think that (a) that excitement is important to effective learning and training; and (b) that our individualization approach contributed to that excitement. However, further analysis of the data from our field tests and possibly further experimentation will be required to confirm our views.

On deciding what works. Where we now stand is that we have a coached practice environment that produces effects of clear economic importance. However, it will take considerable study to determine which of the features of our design worked, which were exercised often enough to even have a chance of being influential, and which, while possibly important in our thinking, need further and more direct demonstration and evaluation. This is the nature of any artifice with utility. It reflects our hypotheses and provides preliminary pilot testing of the validity of those hypotheses. It is, in consequence, both more and less than the theory that motivated it.

References

Anderson, J. R. (1983). The architecture of cognition. Cambridge, MA: Harvard University Press.

Brown, J. S., Burton, R. R., & De Kleer, J. (1982). Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II and III. In D. Sleeman & J. S. Brown (Eds.), Intelligent tutoring systems. London: Academic Press.

Collins, A., Brown, J. S., & Newman, S. E. (In press). Cognitive apprenticeship: Teaching the craft of reading, writing, and mathematics. In L. B. Resnick (Ed.), Knowing and learning: Issues for a cognitive science of instruction. Hillsdale, NJ: Lawrence Erlbaum Associates.

Jordan, B. (1987). Modes of teaching and learning: Questions raised by the training of birth attendants. Technical Report #004. Palo Alto, CA: Institute for Research on Learning.

[18] AML was on an enquiry commission of the Rothschild Foundation and the Ministry of Education and made the observations reported while looking at a printed reading curriculum developed by the Center for Educational Technology in Ramat Aviv, to whom he is now an advisor.

Lesgold, A., Lajoie, S., Logan, D., & Eggan, G. (In press). Applying cognitive task analysis and research methods to assessment. In N. Frederiksen, R. Glaser, A. M. Lesgold, & M. Shafto (Eds.), Diagnostic monitoring of skill and knowledge acquisition. Hillsdale, NJ: Lawrence Erlbaum Associates.

Lesgold, A. M., Lajoie, S. P., et al. (April, 1986). Cognitive task analysis to enhance technical skills training and assessment. Technical Report. University of Pittsburgh, the Learning Research and Development Center.

Nichols, P., Pokorny, R., Jones, G., Gott, S. P., & Alley, W. E. (In press). Evaluations of an avionics troubleshooting tutoring system. Special report. Brooks AFB, TX: Air Force Human Resources Laboratory.

Rogoff, B., & Lave, J. (Eds.) (1984). Everyday cognition: Its development in social context. Cambridge, MA: Harvard University Press.

Terrace, H. S. (1964). Wavelength generalization after discrimination learning with and without errors. Science, 144, 78-80.

Acknowledgements

Sherlock was a massive undertaking. It took a lot of work from a lot of people. It was, as a real-world applied activity in a setting with real work that could not be halted, a test not only of ingenuity and hard work but also of courage, patience, tolerance, perseverance, and the love and respect of many players for each other. Joining us in the effort at one point or another were Jaya Bajpayee, Jeffrey Bonar, Lloyd Bond, James Collins, Keith Curran, Richard Eastman, Drew Gitomer, Robert Glaser, Bruce Glymour, Linda Greenberg, Denise Lensky, Debra Logan, Maria Magone, Tom McGinnis, Valerie Shalin, Cassandra Stanley, Arlene Weiner, Richard Wolf, and Laurie Yengo. Equally important were Sherrie Gott and the many people who worked with her at the Air Force Human Resources Laboratory, Dennis Collins and other Air Force subject matter experts who helped us when we needed it most, and the officers and personnel of the 1st and 33rd Tactical Fighter Wings, at Langley and Eglin Air Force Bases, especially those in the Component Repair Squadrons, who made room in their worlds for Sherlock and his many retainers.

The work was done as part of subcontracts with HumRRO and Universal Energy Systems, who were prime contractors with the Air Force Human Resources Laboratory. Many of the tools and much of the infrastructure that permitted us to efficiently conduct this project were put in place by earlier efforts and the foresight of the Office of Naval Research, where Henry Halff, Susan Chipman, Marshall Farr, and Michael Shafto patiently invested in a future that took a number of years to unfold.

Table 1: Abstracted Problem Space Object Structure

InstanceName - What this hode will be called in action menus and submenus.

Message - Printed in the Concept Window when the student enters the node. If this node is a part of the problem space that it doesn't make sense to search, this message may amount to a hint that this is the wrong place to look.

Display - Which view of the work environment should be displayed when this node is entered; defaults to leaving current display in window.

Preconditions - Conditions to be met upon entry to the node (for example, if this node represents swapping a drawer of the test station, that drawer must be turned off before the swap can occur.

Options - List of nodes to be included in the actions menu for this node; i.e., what can come next?

Parents - Backward pointers in abstracted problem space graph structure.

Children - Forward pointers.

CurriculumSubGoals - List of relevant curriculum issues for this node. These issues are used to calculate the student performance expectation for the node and therefore control the hints that the student gets.

Marker - Indicates whether the trainee completed the search actions subsumed by this node. Can also contain additional information about circumstances in which entering this node is nonproductive or inappropriate, including messages for display in those circumstances.

Hints - Hints for the node (prewritten hints; additional hinting is generated dynamically). In a plan node this is a simple list which are presented in turn to the student. In an actions node, there are four categories of hints and as many as four hints for each category.

Outcome - PASS, FAIL or in some nodes a list of conditions which determine outcome of measurement/test actions subsumed by this node.

NumberOfHintsUsed - Keeps track of the number of hints used (in the action nodes, by category).

LastHintUsed - The number of the last hint used (in action node, by category).

WhoCalledMe - Node the trainee entered from.

WhereDidIGo - Node the trainee exits to.

NodeHistory - List of student actions in the node with time stamps.

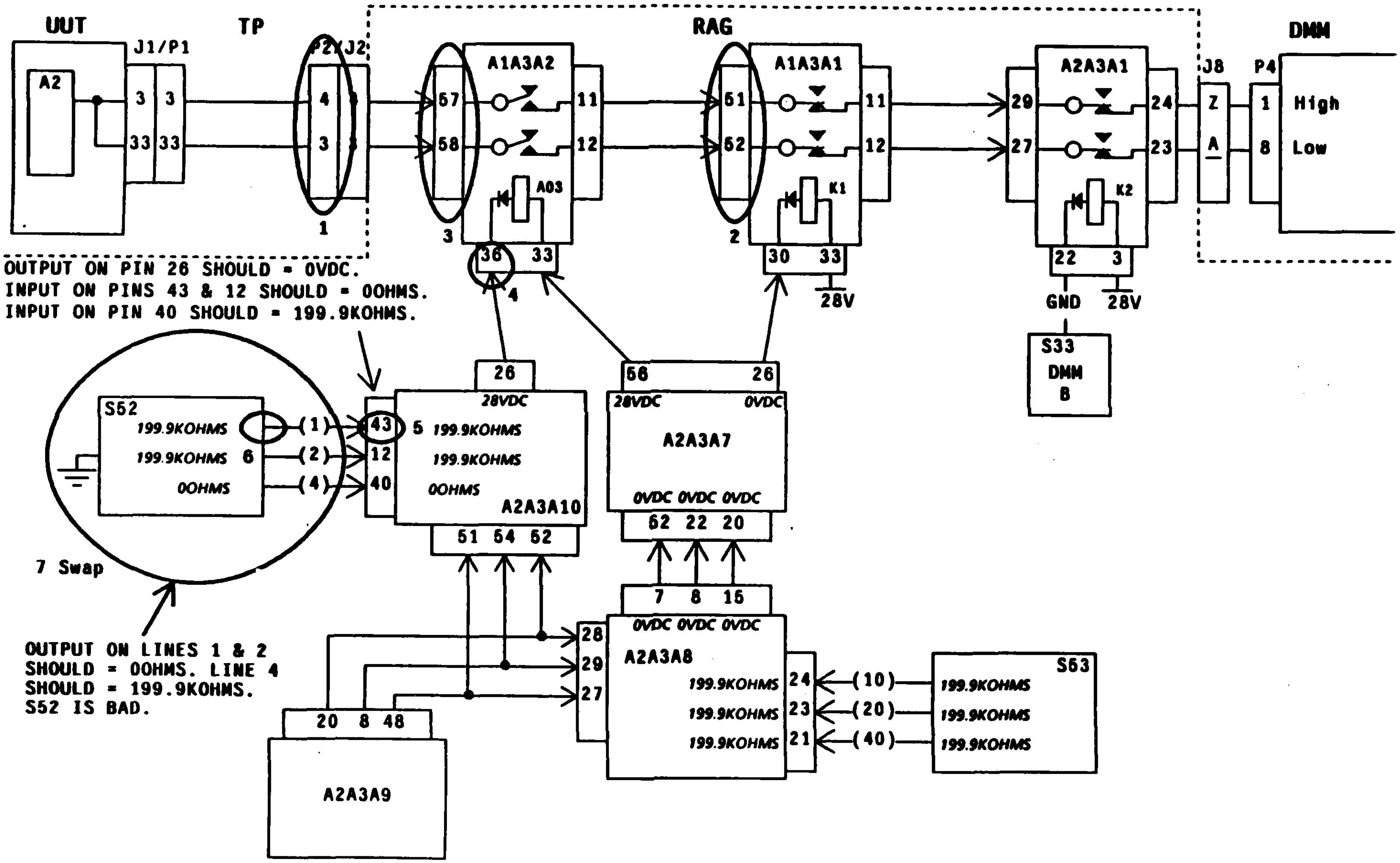
StudentPerfRating - Actual (as opposed to predicted) student performance rating at the node.

StudentPerfExp - Expected student performance at this nooe.

LevelInNode - Controls the category of hint given (e.g., taking measurement actions, interpreting outcomes, and deciding what to do next).

RunTOTestFlg - T if the student needs to rerun the TO test (the activity of the test station that failed because the test station was broken) at this node.

SequenceNumber - Sequence number for order of node entry by trainee.



OUTPUT ON PIN 26 SHOULD = 0VDC.
INPUT ON PINS 43 & 12 SHOULD = 0OHMS.
INPUT ON PIN 40 SHOULD = 199.9KOHMS.

OUTPUT ON LINES 1 & 2
SHOULD = 0OHMS. LINE 4
SHOULD = 199.9KOHMS.
S52 IS BAD.

Figure 1

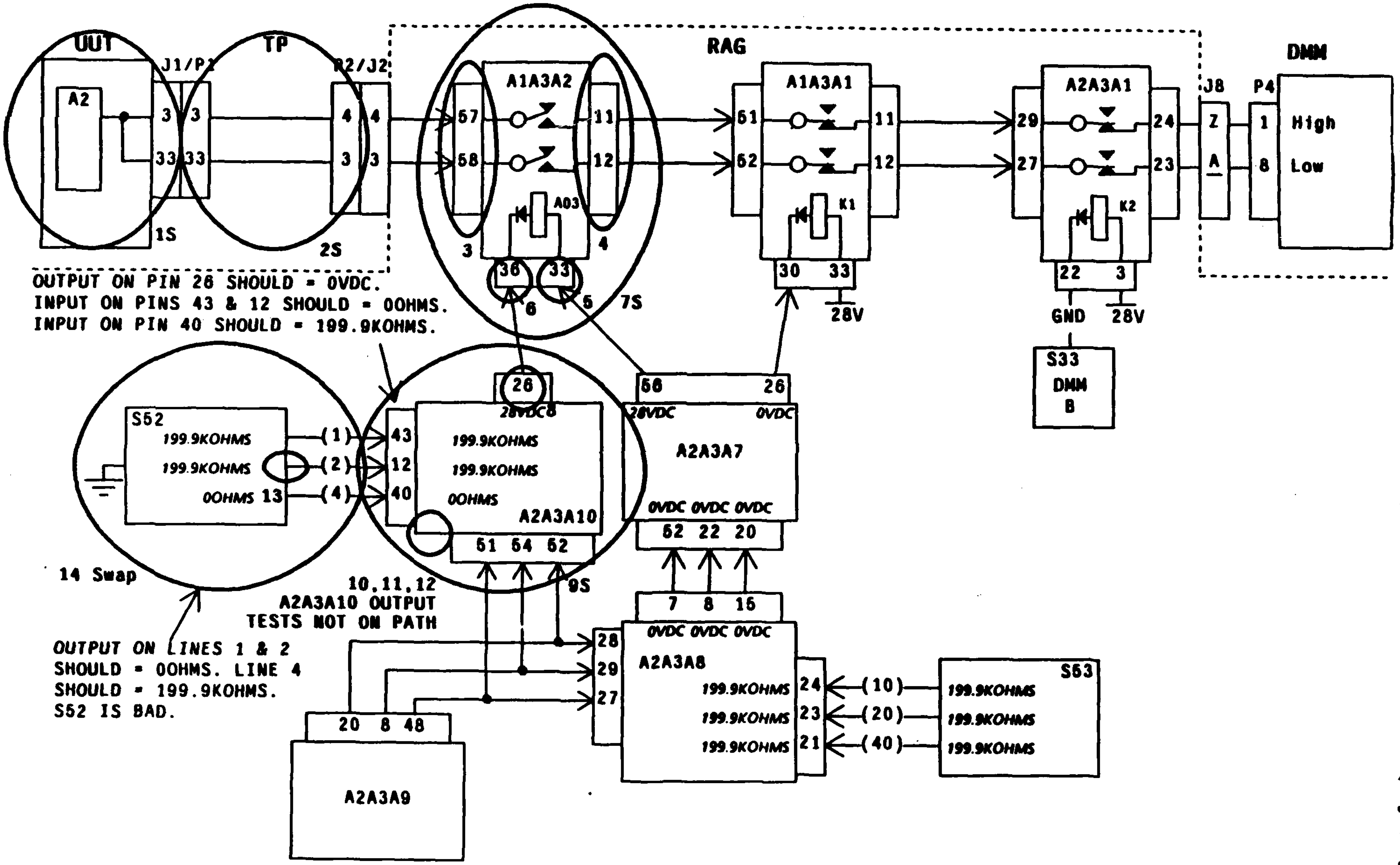

UUT
A2
J1/P1
3
33
1S
TP
P2/J2
4
3
2S
RAG
A1A3A2
57
58
11
12
A03
36
33
3
4
6
5
7S
A1A3A1
61
62
K1
30
33
28V
A2A3A1
29
27
24
23
K2
22
3
GND
28V
S33
DMM
B
J8
Z
A
P4
1
8
DMM
High
Low
OUTPUT ON PIN 26 SHOULD = 0VDC.
INPUT ON PINS 43 & 12 SHOULD = 0OHMS.
INPUT ON PIN 40 SHOULD = 199.9KOHMS.
S52
199.9KOHMS
199.9KOHMS
0OHMS 13
(1)
(2)
(4)
43
12
40
26
28VDC8
199.9KOHMS
199.9KOHMS
0OHMS
A2A3A10
51 54 52
9S
14 Swap
10,11,12
A2A3A10 OUTPUT
TESTS NOT ON PATH
OUTPUT ON LINES 1 & 2
SHOULD = 0OHMS. LINE 4
SHOULD = 199.9KOHMS.
S52 IS BAD.
56
26
28VDC
0VDC
A2A3A7
0VDC 0VDC 0VDC
52 22 20
7 8 15
0VDC 0VDC 0VDC
A2A3A8
28
29
27
199.9KOHMS 24
199.9KOHMS 23
199.9KOHMS 21
(10)
(20)
(40)
S63
199.9KOHMS
199.9KOHMS
199.9KOHMS
20 8 48
A2A3A9

Figure 2

Window Image

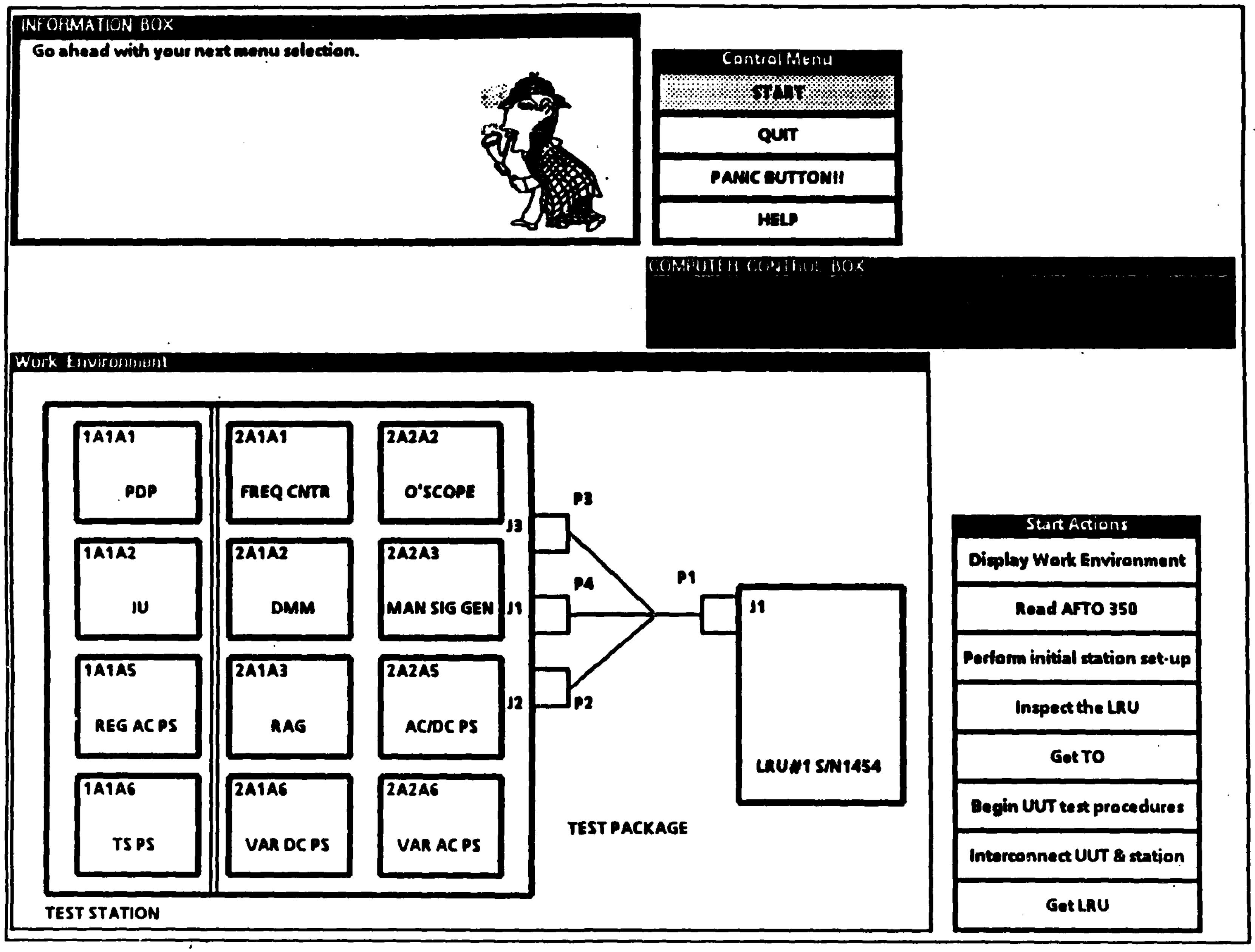

Go ahead with your next menu selection.

Figure 3

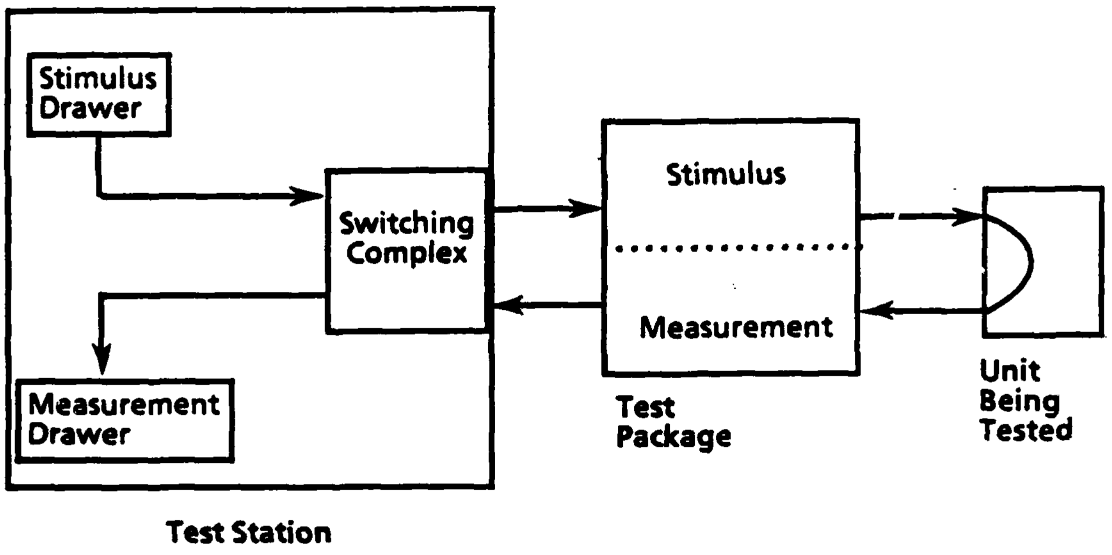

Stimulus Drawer
Switching Complex
Measurement Drawer
Test Station
Stimulus
Measurement
Test Package
Unit Being Tested

Figure 4

Window Image

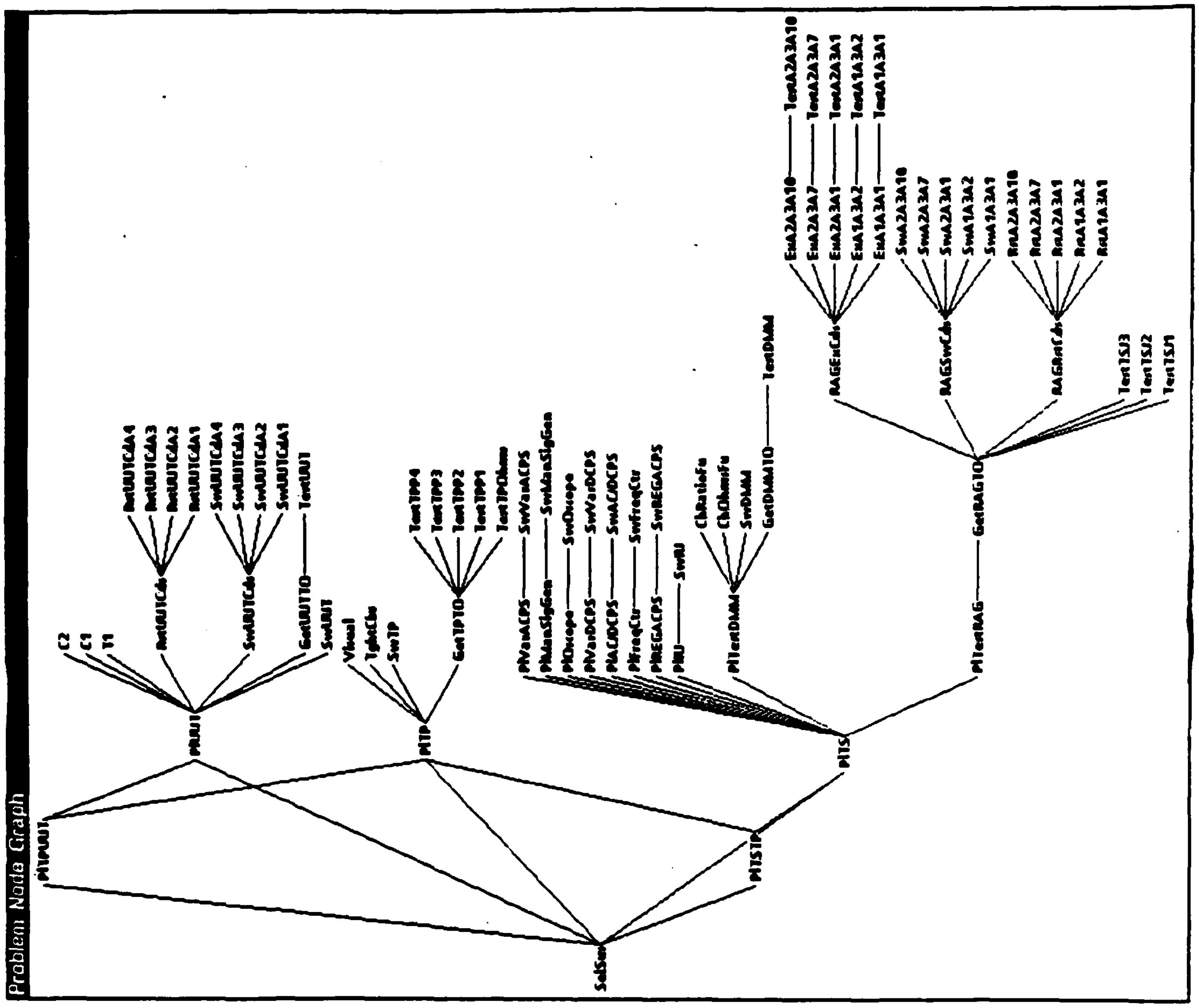


Figure 5

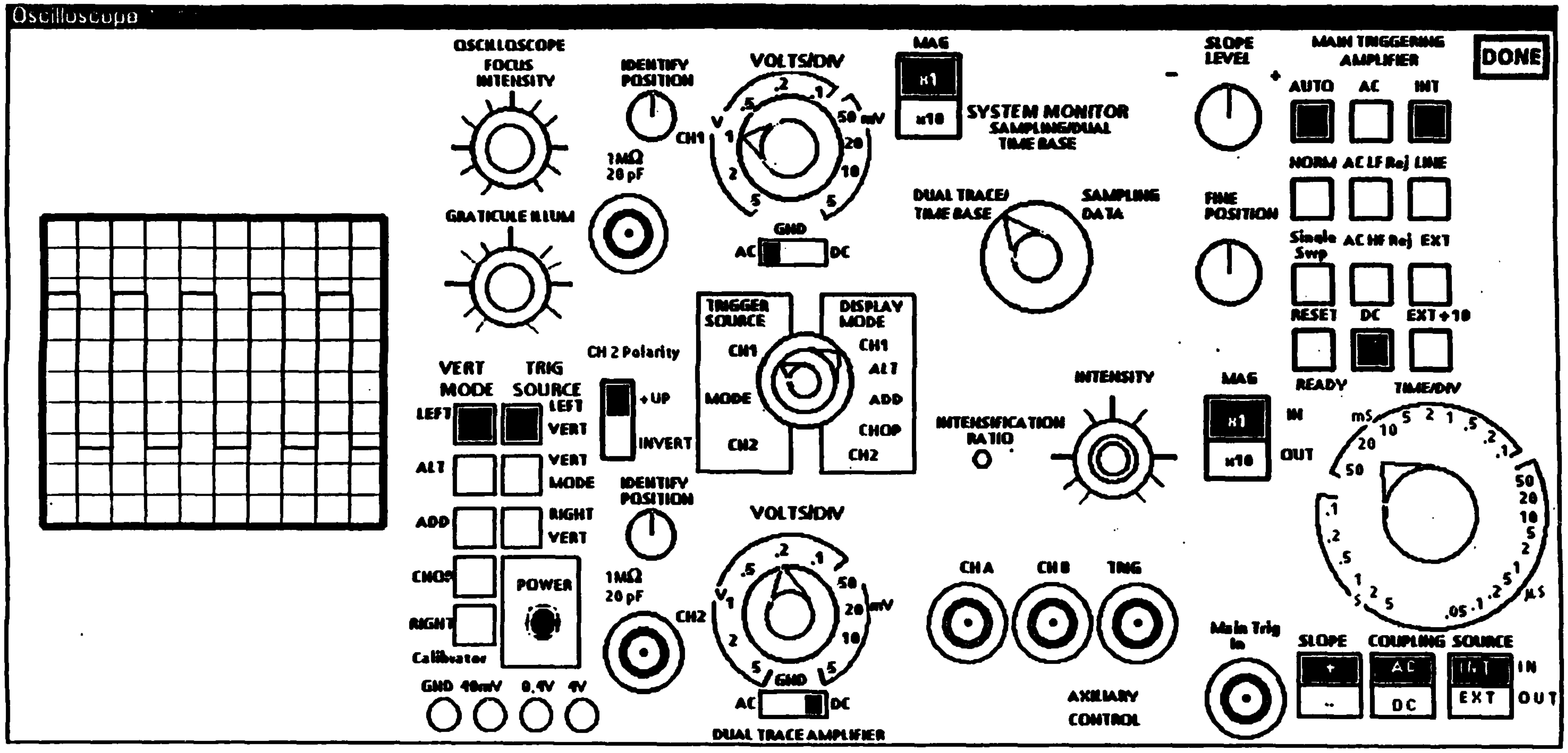

Oscilloscope
OSCILLOSCOPE FOCUS INTENSITY
GRATICULE ILLUM
IDENTIFY POSITION
CH1
1MΩ 20 pF
VOLTS/DIV
V 1 .5 .2 .1 50 mV 20 10 5 2 5
GND AC DC
MAG x1 x10
SYSTEM MONITOR SAMPLING/DUAL TIME BASE
DUAL TRACE/ TIME BASE
SAMPLING DATA
SLOPE LEVEL
- +
FINE POSITION
MAIN TRIGGERING AMPLIFIER
AUTO AC INT
NORM AC LF Rej LINE
Single Swp AC HF Rej EXT
RESET DC EXT÷10
READY TIME/DIV
DONE
VERT MODE
TRIG SOURCE
LEFT
ALT
ADD
CHOP
RIGHT
LEFT VERT
VERT MODE
RIGHT VERT
POWER
Calibrator
GND 40mV 0.4V 4V
CH 2 Polarity
+UP
INVERT
TRIGGER SOURCE CH1 MODE CH2
DISPLAY MODE CH1 ALT ADD CHOP CH2
IDENTIFY POSITION
1MΩ 20 pF
CH2
DUAL TRACE AMPLIFIER
INTENSIFICATION RATIO
INTENSITY
MAG x1 x10 IN OUT
mS 50 20 10 5 2 1 .5 .2 .1 50 20 10 5 2 1 .5 .2 .1 .05 µS
CH A CH B TRIG
AXILIARY CONTROL
Main Trig In
SLOPE + -
COUPLING AC DC
SOURCE INT EXT
IN OUT

Figure 6

Window Image

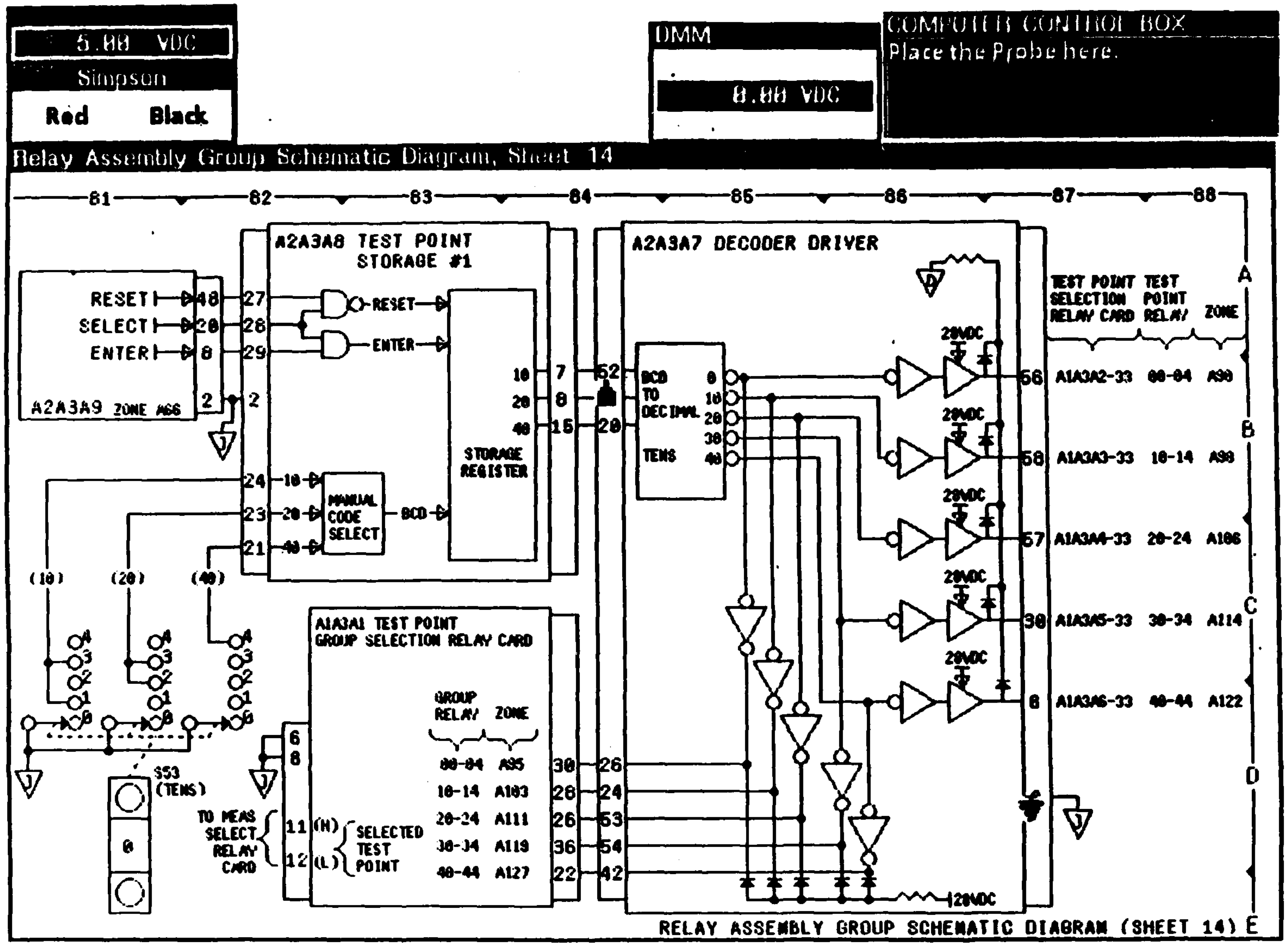
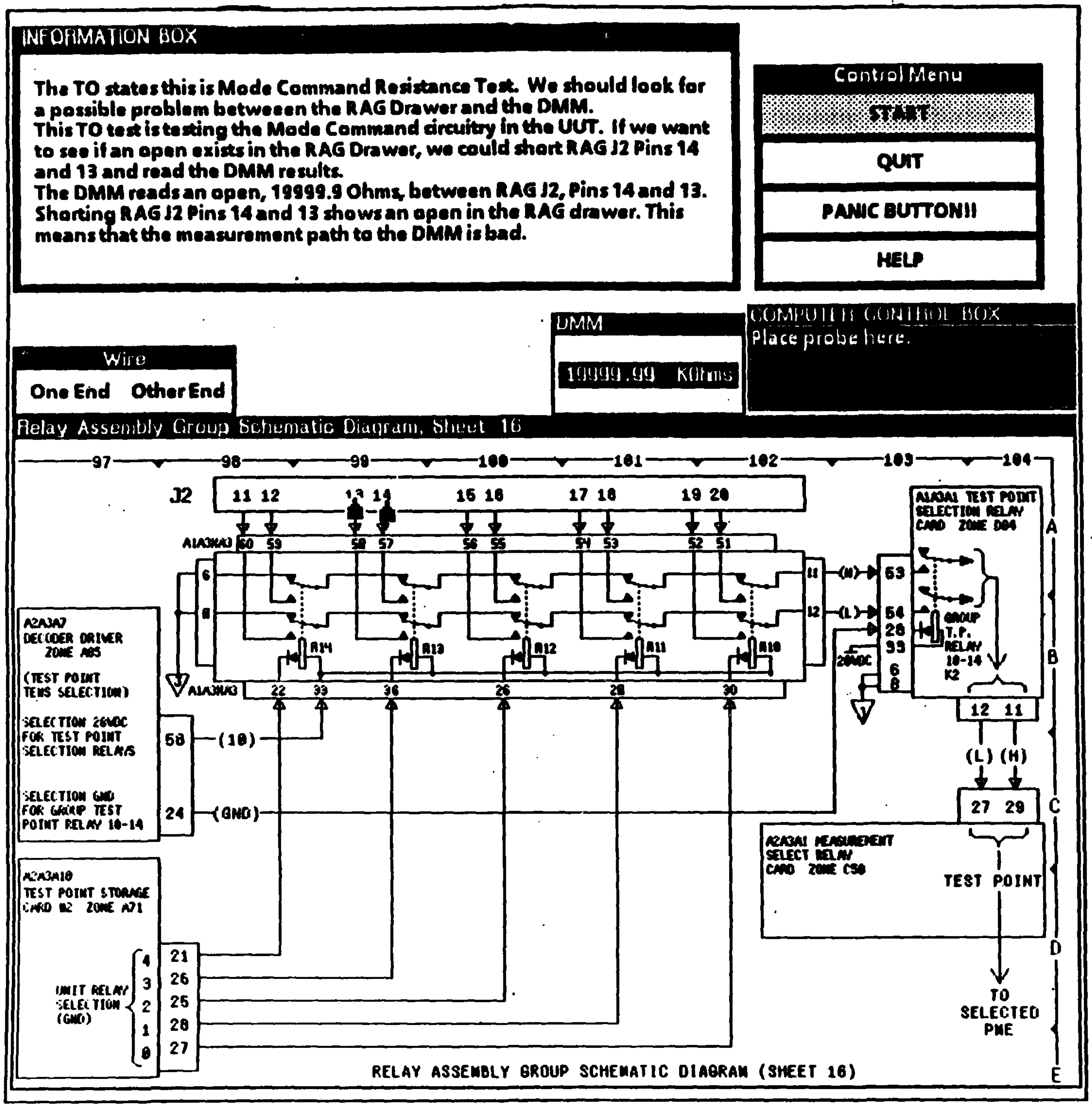


Figure 7

Window Image



INFORMATION BOX

The TO states this is Mode Command Resistance Test. We should look for a possible problem between the RAG Drawer and the DMM.
This TO test is testing the Mode Command circuitry in the UUT. If we want to see if an open exists in the RAG Drawer, we could short RAG J2 Pins 14 and 13 and read the DMM results.
The DMM reads an open, 19999.9 Ohms, between RAG J2, Pins 14 and 13. Shorting RAG J2 Pins 14 and 13 shows an open in the RAG drawer. This means that the measurement path to the DMM is bad.

Control Menu

START

QUIT

PANIC BUTTON!!

HELP

DMM

19999.99 KOhms

COMPUTER CONTROL BOX

Place probe here.

Wire

One End Other End

Relay Assembly Group Schematic Diagram, Sheet 16

97
98
99
100
101
102
103
104
J2
11 12 13 14 15 16 17 18 19 20
A1A3KA3
60 59 58 57 56 55 54 53 52 51
K14 K13 K12 K11 K10
22 33 36 26 28 30
A2A3A7 DECODER DRIVER ZONE A85
(TEST POINT TENS SELECTION)
SELECTION 26VDC FOR TEST POINT SELECTION RELAYS
SELECTION GND FOR GROUP TEST POINT RELAY 10-14
58 (10)
24 (GND)
A2A3A10 TEST POINT STORAGE CARD #2 ZONE A71
UNIT RELAY SELECTION (GND)
4 3 2 1 0
21 26 25 28 27
A1A3A1 TEST POINT SELECTION RELAY CARD ZONE D04
(H) 53
(L) 54
28
33
6
8
26VDC
GROUP T.P. RELAY 10-14 K2
12 11
(L) (H)
27 29
A2A3A1 MEASUREMENT SELECT RELAY CARD ZONE C50
TEST POINT
TO SELECTED PME
A
B
C
D
E
RELAY ASSEMBLY GROUP SCHEMATIC DIAGRAM (SHEET 16)

Figure 8